Scientia Iranica D (2011) 18 (6), 1450–1459

Sharif University of Technology

Scientia Iranica

Transactions D: Computer Science & Engineering and Electrical Engineering

www.sciencedirect.com

Research note

Codebook appearance representation for vehicle handover across disjoint-view multicameras

E. Shabaninia [a], **Sh. Kasaei** [b,*]

[a] *Department of Computer Engineering, Faculty of Engineering, Shahid Bahonar University of Kerman, Kerman, Iran*
[b] *Department of Computer Engineering, Sharif University of Technology, Tehran, P.O. Box 11157-9517, Iran*

Received 10 January 2010; revised 9 February 2011; accepted 6 August 2011

KEYWORDS
Disjoint-views;
Multicameras;
Major color representation;
Codebook model;
Vehicle monitoring systems;
Surveillance.

Abstract Object handover as continually tracking an object across disjoint-view cameras is a necessary part of video-based monitoring systems. While having nonoverlapping cameras is a requirement for monitoring a wide area, there is no common 3D location that can be used to detect multiple views of the same object, in contrast with overlapping cameras. Appearance features play an important role for object handover in such camera networks. This paper focuses on modeling appearance features of moving vehicles by a new major color representation called codebook representation. Toward this end, in each frame, the *k*-means algorithm is used to cluster major colors of an object. In the subsequent frames, a set of cylinders in the RGB space called codebook keeps the track of these major colors for incremental clustering. Then, in the matching phase, a similarity measurement for comparing different codebook sets discriminates major colors of observations. In addition, a brightness transfer function is developed for mapping cylinders between two camera views. By this mapping, the model can tolerate the illumination change of environments. The method is fast enough to be used in real-time applications. Experimental results show the efficiency of the proposed methods on real datasets.

1. Introduction

Widespread use of video cameras, as powerful sensors for collecting visual data in surveillance systems, has provided the ability of monitoring wide areas. In addition, cameras provide information that is conceivable by human operators. By the extend of the number of cameras in surveillance systems, the role of computer vision for automating the process of extracting information from videos has become more important. Understanding the behavior of objects is mostly dependent on tracking objects in a network of cameras. Cameras, depending on the application in mind, may have overlapped or nonoverlapped field of view. The overlap in camera views is helpful for solving the occlusion and depth estimation problems. However, the use of nonoverlapped cameras is usually unavoidable especially in wide areas due to cost and maintenance problems.

Object handover, as moving across disjoint camera views, is a challenging task and many factors including the changes in illumination conditions, different viewing angles, shadows, and environmental noise can introduce major challenges in its process. Since in independent camera views the camera calibration and 3D locations or geometry of cameras cannot be used for detecting multiple views of the same object, appearance and space/time features play an important role in object handover.

In this paper, we present an efficient appearance model for tracking vehicles in a network of disjoint-view cameras. This method can be used for any object with limited number of color clusters (such as humans) that is fast and suitable for real-time applications. The reminder of this paper is organized as follow. Section 2 reviews some related work on multicamera tracking. In Section 3, the proposed method is explained in detail. Performance analysis of the proposed technique is presented in Section 4. Finally, the paper conclusion is derived in Section 5.

* Corresponding author.
E-mail addresses: eshabaninia@gmail.com (E. Shabaninia), skasaei@sharif.edu (Sh. Kasaei).

1026-3098 © 2012 Sharif University of Technology. Production and hosting by Elsevier B.V. Open access under CC BY license.
Peer review under responsibility of Sharif University of Technology.

doi:10.1016/j.scient.2011.11.007

Production and hosting by Elsevier

Table 1: Several multicamera tracking methods with disjoint views.

Method	Features for object handover			Modeling	Handling illumination change
	Appearance	Space/time	Neighboring relationships		
[11]	Edge images	×	×	Probabilistic	×
[12]	Major color spectrum histogram representation (MCSHR)	×	×	Similarity measurement	Intensity transformation
[13]	Mean HSV color and size of object (modeling by multivariate Gaussian density function)	Time, lane, velocity (modeling by some multivariate Gaussian density function)	×	Probabilistic	Supervised learning of parameters and online forgetting update
[14]	Histogram	Time, location, velocity (modeling by KDE)	×	Maximum a posteriori (MAP) estimation	Supervised learning of subspace of brightness transfer functions using PCA
[15]	Mean HSV color	Time, location, velocity (modeling by KDE)	Appearance similarity of neighbors of two observations	MAP estimation	Supervised learning of parameters and online forgetting
[16]	Histogram	Location, time	×	MAP estimation by converting the problem into a linear program	×
[17]	Histogram	Entry/exit zone (by modeling the transition probability)	×	MAP estimation	Unsupervised learning of subspace of brightness transfer functions using PCA
[18]	Bag-of-visterms (a histogram of quantized local feature descriptors)	×	×	Linear kernel SVMs	Learn + MT algorithm

2. Related work

In literature, the task of visual surveillance is divided into several stages [1–3]. According to [1], the whole task of visual surveillance can be divided into two main subtasks of single camera and multicamera tracking.

Although many reports are available for tracking objects in the single camera phase [4–7], there are just few that address the multicamera tracking especially with nonoverlapped field of view. Most overlapping multicamera tracking methods use camera calibration and 3D locations [8–10] and thus cannot be used for independent views due to space/time distance between cameras. The methods related to nonoverlapped cameras often try to model features in different cameras and then propose a similarity measure (or a probabilistic method) for data association. Several methods of multicamera tracking with disjoint views are listed in Table 1.

Features used for multicamera tracking are categorized in three main classes of appearance, space/time, and neighbors' relations. Appearance is the main feature for multicamera object handover. In [11], Shan et al. presented an unsupervised learning approach for measuring the frame edges and matching the appearance between nonoverlapping views. The matching is based on computing the probability of two observations from different views. As that method compares the edge images of vehicles, the images should be registered first. In [12], Madden et al., proposed an algorithm for modeling the *major color spectrum histogram* (MCSHR) of tracking objects based on an online *k*-means color clustering algorithm (a well-known clustering algorithm). By incremental use of frames and a data adaptive intensity transformation they have compensated for deformable objects and illumination changes. A similarity measurement has been also introduced to compare the appearance representations of any two arbitrary individuals. The usage of Bayesian framework for nonoverlapped camera tracking initiates from [13,16].

In [13], Huang and Russell defined a physical event space over which probabilities were defined. Then, by introducing an identity criterion they were able to compute the probability that any two given objects are the same, given a stream of observations of many objects. They used both appearance and space/time features. In [16], Kettnaker and Zabih introduced Bayesian framework for multicamera surveillance task and showed how the MAP solution can be found under some additional independence assumptions by transforming the problem into a compact linear program. Appearances of objects were represented using histograms. Makris et al. [19] determined the topology of a camera network by determining the entry and exit zones of each camera and the links between these zones by using the co-occurrence of entry and exit events. The method proposed that correct correspondences can cluster the feature space (location and time). In [14], Javed et al. proposed an algorithm for tracking objects in a network of nonoverlapping cameras. The algorithm learned the camera topology in the form of multivariate probability density of space/time variables by using KDE. It also learned the subspace of inter camera *brightness transfer function* (BTF) to handle the appearance change of an object as it moves from one camera to another. In [17], Chen et al. proposed an unsupervised method which learns both spatio-temporal and appearance relationships adaptively and thus can be applied to long-term monitoring.

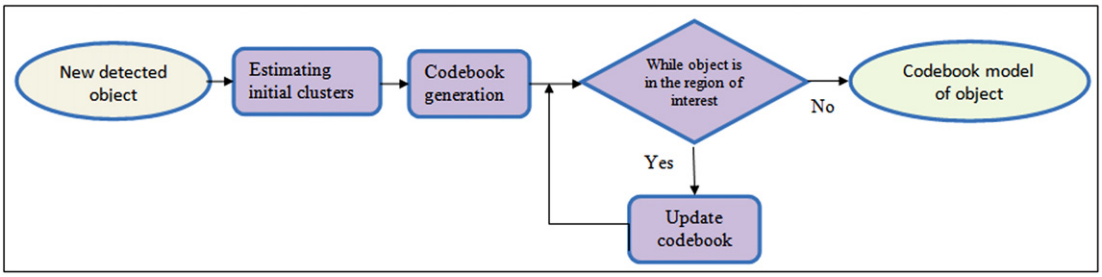

Figure 1: Block diagram of the proposed method for generating Codebook of a detected object.

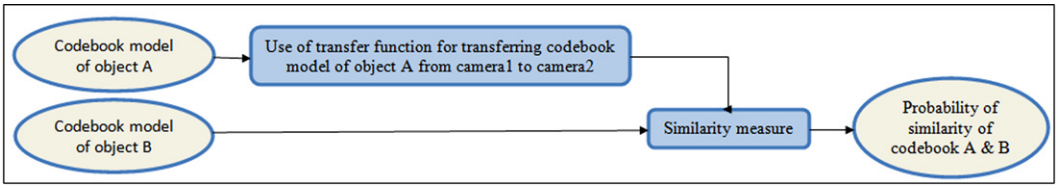

Figure 2: Block diagram of the proposed method for calculating the probability of similarity of two Codebooks.

In [18], Teixeira et al. proposed bag-of-visterms (*i.e.,* a histogram of quantized local feature descriptors) to represent and match tracked objects. Incremental (or adaptive) learning was used to tackle changes of objects over time. However, the method intrinsically comes from image retrieval applications and needs a large descriptor size.

In our previous work [15], we modeled the appearance of vehicles by the mean HSV color of separated object and Parsen window was used for modeling space/time features. To improve the tracking accuracy, the similarity of neighboring vehicles was encountered. After learning the parameters of estimation, correspondent vehicles were determined in a MAP framework.

In this paper, we focus on modeling the appearance of vehicles for multi independent camera tracking, along with a similarity measure that can be used independently by any feature model in a Bayesian (or any other) probabilistic framework. In contrast to appearance models reported so far (that are mostly histogram-based), our model uses the major colors of each object during the tracking process. Histograms principally work well for still images (because a small perturbation in the second frame may drastically change the histogram of the object even in the same camera view). Consequently, it is needed to develop a new model for videos and moving objects. Here, we propose to consider the track of major colors of an object and make our model robust by tolerating any perturbation that occurs along the illumination axis. Since the number of major colors is usually limited, the matching process can be performed very fast.

3. Proposed object appearance representation method

Suppose we have found track of objects in each camera. In [20], we proposed a method for single camera tracking. Passing a vehicle, a person, or any object across the view of a camera represents a list of features for that observation. That can be the pixel color, size, 3D model, edge, location (x, y), time, velocity, neighboring observations, or any other feature. Among this long list, the pixel color, size, 3D models, and edges are usually called appearance characteristics (or features) of an observation. In this paper, we use the color of objects as appearance representation and concentrate on modeling major colors of an observation. Major colors are a set of colors where most pixels of an object have exact (or very similar) colors with those presented in that set. This definition can be given more technically in terms of a set of color clusters of an object where the sum of their probabilities exceeds a threshold. In fact, for object a in frame i, major colors are defined by

$$MajorColors\,(a) = \{c_1,\ c_2, \ldots, c_n\} \quad (1)$$

such that $p(c_1) + p\,(c_2) + \cdots + p(c_n) \geq$ Threshold.

In [14,17,16], color clusters are histogram bins with *Threshold* $=$ 1 (so $p(c_1) + p\,(c_2) + \cdots + p(c_n) = 1$). In [12], they are a group of bins that form a relatively large number of simple spherical clusters with the same radius. In our method, clusters are cylinders in the RGB space that grow with time. This growing is along the illumination axis. By capturing the most probable centers of clusters in a cylinder in time, we can predict the exact centers of clusters of an object. The main steps of our approach are as follow. (I) A heuristic method for estimating initial number of clusters of an object in the first frame is used. (II) Then, this initial estimate is used as the parameters of the k-means algorithm for clustering the object color in subsequent frames. Each set of colors, in each frame, forms a set of cylinders in the RGB space, called codebook. (III) After computing the object codebook for each frame in its track, these codebooks are integrated with the known previous codebook (obtained from previous frames). Therefore, given a history of measurements for a system, a model is built for the state of the system that maximizes *a posteriori* probability of those previous measurements. (IV) In order to compensate for differences in the global illumination of different camera views, we use a learning method for a map function. (V) Finally, a similarity measure is proposed for comparing two sets of codebooks for two observations of cameras. Figures 1 and 2 show the block diagram of the proposed method.

3.1. Initial set of clusters

In the first step of the algorithm, the initial set of clusters is created. In the first frame, the object pixels are scanned in a row order. As the first pixel appears, its color is set as the centre of the first cluster. If each following pixel stays within a threshold from an existing cluster centre in Euclidean distance, the pixel count for that cluster is increased by one and otherwise a new

cluster, centered on that pixel, is created. In the RGB space, this is equivalent to have uniformly spaced clusters with a common radius. This procedure is similar to that proposed by Madden et al. [12] and Li et al. [21] to calculate the principal colors.

After calculating the initial number of clusters, we use the *k*-means algorithm to compensate for significant displacement of cluster centers (that may occur because of the simple initial cluster creation procedure). The *k*-means [22] is a very simple and effective clustering algorithm, which requires specifying the number of sought clusters in advance. To initiate the cluster centers, the *K* points are chosen at random and the data measurements are assigned to their closest cluster center. Then, in each cluster, the mean of all data points is calculated and considered as the new centroid of that cluster. The whole process is iterated with the new cluster centers. This iterative process continues until convergence. Note that it might converge to a local minimum and different final cluster centers can arise by choosing different initializations. But, by the intuition of our heuristic method for initial clusters it might reach into the global minima. By iteration, changes in centroid positions tend to decrease gradually. The final set of clusters forms our initial set of cylinders in the RGB space, called codebook.

3.2. Codebook representation for incremental clustering

In [23], Kim et al. proposed a representation, called codebook, for modeling the background of a scene in order to classify moving objects by a background subtraction approach. As they proposed, a codebook is in fact a set of cylinders in the RGB space where each cylinder is called codeword that tends to grow along the illumination axis. To deal with global and local illumination changes (such as shadows and highlights), they developed a color model to perform a separate evaluation of color distortion and brightness distortion rather than normalized colors that many algorithms might use. The motivation of this model is the empirical observation that background pixel values are mostly distributed in elongated shape along the axis passing toward the origin point; since the variation is mainly due to brightness.

In their method, for each background pixel, a collection of codewords is considered that represents its different conditions. For example, consider a scene with green tree leaves fluctuating over a yellow wall. A yellow wall pixel becomes green when a leaf passes over it. Note that both yellow and green values represent the background. In the codebook method, two codewords (cylinders) for this pixel are considered. The collection of codewords for a pixel forms the pixel codebook.

In this work, we use codebooks in a completely different manner. We employ codebook representation for modeling variations of major color clusters of an object. Once we have calculated the initial set of clusters from previous section, a *codebook* (CB) is created for that object, as

$$CB = \{cw_1, cw_2, \ldots, cw_n\} \tag{2}$$

Where cw_i denotes the ith codeword and n is the number of clusters in the initial set. The cw_i represents a cylinder in the RGB space, as

$$cw_i = \left\{v_i, \hat{I}_i, \check{I}_i, f_i, p_i\right\} \tag{3}$$

where v_i is the $\{R, G, B\}$ triple denoting the center of the ith cluster. $\hat{I}_i$, $\check{I}_i$ represent the maximum and minimum brightness values of all pixels assigned to that codeword, respectively.

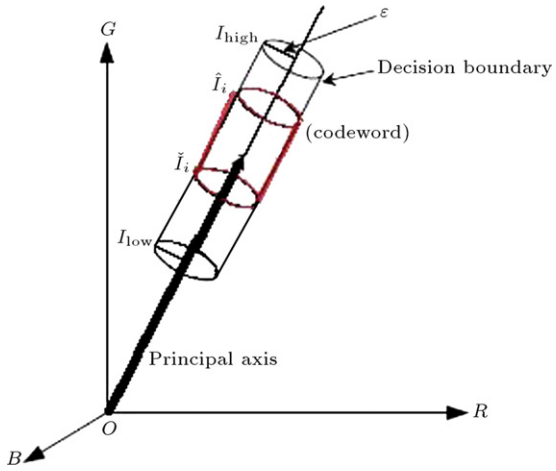

Figure 3: The codeword representation, with separate evaluation of color and brightness distortion (original image courtesy of [23]).

Remaining are f_i, the frequency of activation of that codeword, and p_i, denoting the probability of that codeword. It is set to the ratio of the number of pixels assigned to that cluster to the total number of object pixels.

A codeword is called activated when a point falls in the decision boundary of that codeword. The decision boundary is an enclosing cylinder with the same radius and center but with a slightly wider bound. This is to allow pixels with varying illuminations to have a wider range of minimum and maximum brightness values, defined by

$$I_{low} = \propto \hat{I}_i, \qquad I_{hi} = \min\left\{\beta\hat{I}_i, \frac{\check{I}_i}{\propto}\right\} \tag{4}$$

where $\propto < 1$ and $\beta > 1$, as defined in [23]. Figure 3 shows a *cw* and its decision boundary; that is used for incremental clustering. In our modeling, for simplicity, the radius of cylinders is considered as a constant, called ε.

In subsequent frames, the *k*-means algorithm is used with the same number of clusters found in the first frame. This gives the major color clusters in each frame. Now, we can update the initial codebook to integrate new values of cluster centers. The update process is a combination of previous measurements and the new ones. It tries to maximize the *a posteriori* probability of previous measurements by averaging the parameters of a codeword with the new available information. The pseudo-code of the update process (incremental clustering) is depicted in Figure 4. The input to the process is the codebook of the object in the new frame. *Cluster_centroids* and *cluster_probs* are two vectors representing the new centers of clusters and their probabilities; resulting from the *k*-means with the same number of clusters in the new frame. As this figure shows, each new value of centers is first compared with the previous codewords of the object. If it lies inside the decision boundary of any existing cylinders (*i.e.*, there is a match), that codeword is updated. If no match is found, a new codeword is created. In this process, a codeword becomes updated if a new cluster center in the new frame lies within the decision boundary of the codebook (*i.e.*, when the color distortion of the new center is less than ε and its brightness lies in the range of $[I_{low}, I_{hi}]$).

3.3. Migration from one camera view to another

Since nonoverlapping camera views usually have space/time differences, the appearance of objects can change drastically

```
Incremental_process ( clusters_cesntriods[] , cluster_prob[])
  {
  For (t=1 to N) {
     //N is the length of clusters_cesntriods [] and cluster_prob []
     (R, G, B) = clusters_cesntriods[t] ;
     I = √(R² + G² + B²);
      Find codeword cw_m in CB = {cw_1, cw_2, ..., cw_l} such that its center point be in Decision Boundary of cw_m
      If (there is no such codeword (i. e no match))
        {
         //create a new codeword cw_l by setting
           L ← L + 1 ; v_l ← (R, G, B);
          Î_l ← I;  Ǐ_l ← I ;
           p_l ← cluster_prob [t];
        }
      Else
        {
         //update the matched codeword cw_m
           v_m ← ((f_m R_m + R)/(f_m + 1), (f_m G_m + G)/(f_m + 1), (f_m B_m + B)/(f_m + 1));   Î_m ← max (I, Î_m);   Ǐ_m ← min(I, Ǐ_m);
           p_m ← (f_m p_m + cluster_prob [t])/(f_m + 1);
        }
    } // end of For

  }
```

Figure 4: Pseudo-code of the update process (incremental clustering).

from one camera to another. Agents like time varying illumination sources (particularly in outdoor), shadows (especially self-shadows), and deformable objects (like pedestrians) cause major challenges in accurate modeling. Consequently, it is common to have a transformation when migrating from one camera to another. In histogram-based approaches, the use of brightness transfer functions (or BTFs for each color channel) is very popular [14,17,24,25]. This transfer function uses the fact that the percentage of image points in the first observation (Camera 1) with brightness less than (or equal to) B_i is equal to the percentage of image points in the second observation (Camera 2) with brightness less than (or equal to) B_j, when B_i, B_j are histogram bins in normalized histograms of an object in two views.

In this section, we learn a similar transfer function across different views in order to compensate varying illuminations. Learning is done for each pair of cameras by assuming that the correspondence is known. One way to achieve this is manually selecting the tracks of similar objects.

In order to learn the mapping of the same color between different cameras, we used a supervised learning phase in which it is assumed that the correspondence of objects is known. During this phase, each object is modeled in each camera view with only one cluster that is the mean color of that object. This single cluster represents the object by a codebook that contains only one codeword. Then, by tracking this object its codebook becomes updated. As such, two correspondent objects form two cylinders in the RGB space. In this phase, the corresponding cylinders, say cw^1 and cw^2, are used to learn the transfer function between cameras, where the superscripts denote the camera number. For mapping a cylinder whose principal axis passes through the origin of the Cartesian system and preserves its radius and height, it is enough to find only the mapping of principal point. In other words, it is sufficient to find a mapping F, from $v^1 = \{R^1, G^1, B^1\}$ in cw^1 to $v^2 = \{R^2, G^2, B^2\}$ in cw^2 for all correspondent objects in two cameras, as

$$F: R \times G \times B \rightarrow R \times G \times B. \tag{5}$$

Considering each dimension separately, the F function will in fact form three mappings for each dimension that can be estimated by interpolating the known parts of data. By estimating the map function, an approximation of the new location of any given cylinder in the second camera can be found. This method is somewhat similar to cumulative transfer function in [24] with single bin histograms. But, instead of mapping a color to another color we map a cluster of colors in one view to another. This transfer function is updated frequently to compensate for dynamic conditions of environment. This can be performed by adding new matched objects to the model or by following the changes of each view, separately, similar to [25].

3.4. Similarity measure

Codebook representation can be considered as a general method for modeling the appearance of a moving object. It has the capability to cope with varying illuminations. But, since the ultimate goal of multicamera tracking is to find the track of an object, it is necessary to have a similarity measure for the appearance model of objects. This measure is usually expressed in terms of likelihood in order to be used in a probabilistic framework (as in [14]). In this subsection, we introduce a similarity measure for codebook representation that is also fast enough to be computed in real-time applications.

We start by reviewing the information that is available in the test phase. For each camera, there is a list of objects that have been observed in the region of interest of that camera and the appearance of each object is represented by a codebook (a set of cylinders). Now, we are interested to find the track of each object in the whole environment (*i.e.,* to know which object in one camera is mostly similar to a particular object in another camera). As it was mentioned before, this similarity can be in appearance, space/time features, or neighbors' relationships. In this paper, we focus on appearance and thus the problem is to assign a similarity measure between each two codebooks.

Now, denote the codebook of an object in the first camera as $CB^1 = \{cw_1{}^1, cw_2{}^1, \ldots, cw_n{}^1\}$ and the codebook of another object in the second camera as $CB^2 = \{cw_1{}^2, cw_2{}^2, \ldots, cw_m{}^2\}$. First, we map each codeword (or cylinder) in the first set to another using the transfer function F. Then, for transferred CB^1 and CB^2 we define the similarity measure as fallow:

```
Similarity Measure (CB¹ = {cw₁¹, cw₂¹ .... cwₙ¹}, CB² = {cw₁², cw₂² .... cwₘ²})
{
  // it is supposed that CB¹ is the transferred codebook using the transfer function between two cameras
  Similarity =0;
  For (i=1 to n) // for all codewords in CB¹
  {
      S = {all codewords in CB² that their angles with cwᵢ¹ in CB¹ are less than the ThresholdAngle}
      // angle = acos ((vᵢ¹.vⱼ²) / (|vᵢ¹|.|vⱼ²|))
      Find cwⱼ² in S that has the maximum overlap with cwᵢ¹
      maximum-overlap = min(Î¹ᵢ, Î²ⱼ) - max(Ǐ¹ᵢ, Ǐ²ⱼ)
      If (such codeword doesn't exist)
          Continue;
      Else If (maximum-overlap > ThresholdOverlap)
          Similarity = Similarity + min (pᵢ¹, pⱼ²);
  }// end of for
  Return similarity;
}
```

Figure 5: Pseudo-code for calculating the similarity measure.

Figure 6: Location of cameras in Kordestan highway.

For each codeword cw_i^1 in transferred CB^1, find a codeword cw_j^2 in CB^2 that has the maximum overlap with this codeword. If such codeword exists, add $\min(p_i^1, p_j^2)$ *to the probability of similarity of these two objects, where p_i^1 and p_j^2 are the probabilities of cw_i^1 in CB^1 and cw_j^2 in CB^2, respectively.*

In order to fasten the process of finding a cylinder that has the maximum overlap with a given cylinder, codewords for which the angle between their principal axis and the given cylinder axis is less than a threshold (say 10°) are searched. Then, among those cylinders the one for which the minimum and maximum brightness values have the maximum overlap is chosen. Figure 5 shows the pseudo-code of calculating the proposed similarity measure.

The inputs to this process are two codewords and their similarity CB^1 and CB^2 are computed. It is assumed that all codewords in CB^1 are previously transferred; by using the transfer function between two cameras. The *ThresholdOverlap* and *ThresholdAngle* are two thresholds that are used for minimum overlap and maximum angle values. The experimental results showed that *ThresholdOverlap* in $[-15 \cdots 0]$ and *ThresholdAngle* in $[5 \cdots 10]$ degrees yield good results.

Camera 1 Camera 2 Camera 3

Figure 7: Sample images captured by three cameras.

Table 2: Similarity measure for 5 selected vehicles in the first scenario: passing from Camera 1 to 2 without applying the transfer functions.

C1 \ C2	Vehicle1	Vehicle2	Vehicle3	Vehicle4	Vehicle5
Vehicle1	0.71474	0.51563	0.86077	0.45633	0.27566
Vehicle2	0.58666	0.33958	0.65254	0.53449	0.73515
Vehicle3	0.79474	0.46446	0.46446	0.31677	0.62597
Vehicle4	0.72248	0.88476	0.48108	0.88834	0.44378
Vehicle5	0.47919	0.71594	0.63564	0.08124	0.45540

Table 3: Similarity measure for 5 selected vehicles in the second scenario: passing from Camera 2 to 3 without applying the transfer functions.

C2 \ C3	Vehicle1	Vehicle2	Vehicle3	Vehicle4	Vehicle5
Vehicle1	0.9508	0.7584	0.9773	0.6901	0.8846
Vehicle2	0.8145	0.9696	0.5282	0.5396	0.8575
Vehicle3	0.1802	0.5483	0.8194	0.6442	0.4896
Vehicle4	0.8526	0.3688	0.5529	0.4718	0.5992
Vehicle5	0.6947	0.5083	0.6916	0.4336	0.9051

4. Experimental results

As there is no standard database available for evaluating the performance of disjoint-view multicamera trackers, we used our captured sequences of three cameras in Kordestan highway of Tehran. The cameras were placed in a distance of about 500 (m) from each other. Each sequence has a frame rate of 25 frames/second with 576 × 720 pixels/frame. Vehicles move in one direction from north to south. There are some ramps between camera sites. Also, some vehicles appear or disappear in successive cameras. Figure 6 shows the location of these cameras and in Figure 7 some sample images captured by these cameras are illustrated.

To estimate the brightness transfer function, a training phase is required to learn the corresponding vehicles in that environment. In the training phase, it is not needed to find the correspondences of all vehicles, but the number of samples should be high enough to properly estimate the functions. Figure 8 shows the estimated brightness function between Camera 1 and Camera 2 and between Camera 2 and Camera 3, for each RGB channel.

In order to evaluate our proposed object representation model, we managed two scenarios. Passing from Camera 1 to Camera 2 and passing from Camera 2 to Camera 3. We tested our method on automatically segmented and tracked objects. For our purpose, we selected 5 vehicles with relatively distinct colors that were observed in all three cameras. Figure 9 shows a sample view of these vehicles in three cameras along with their automatically segmented masks. As this figure shows, Cameras 1 and 2 do not have a good resolution that causes some problems for the subsequent segmentation process.

As proposed in Section 3.2, we use a codebook representation for each observed vehicle in each camera that consists of a set of cylinders in the RGB space. These cylinders can grow with time along their principal axes in order to compensate for illumination changes of environment. Figure 10 shows the codebook of selected vehicles in three cameras. The results of calculating the similarity measure for these two scenarios are reported in Tables 2–5. Table 2 and 3 show the results without considering the transfer functions between cameras while Tables 4 and 5 show the results after applying the transfer functions.

Table 4: Similarity measure for 5 selected vehicles in the first scenario: passing from Camera 1 to 2 after applying the transfer functions.

C1 \ C2	Vehicle1	Vehicle2	Vehicle3	Vehicle4	Vehicle5
Vehicle1	0.91752	0.65611	0.60440	0.33344	0.38689
Vehicle2	0.74470	0.92181	0.80350	0.54691	0.81620
Vehicle3	0.49231	0.49730	0.82408	0.32130	0.49731
Vehicle4	0.52291	0.18761	0.83240	0.92261	0.46327
Vehicle5	0.73820	0.58340	0.69405	0.46489	0.85355

Table 5: Similarity measure for 5 selected vehicles in the second scenario: passing from Camera 2 to 3 after applying the transfer functions.

C2 \ C3	Vehicle1	Vehicle2	Vehicle3	Vehicle4	Vehicle5
Vehicle1	0.93184	0.75840	0.75841	0.90916	0.64200
Vehicle2	0.81452	0.96968	0.85757	0.90093	0.52828
Vehicle3	0.29320	0.71127	0.98237	0.77871	0.60735
Vehicle4	0.51871	0.36889	0.53449	0.97184	0.35520
Vehicle5	0.69475	0.53531	0.73771	0.57951	0.95115

Comparing the values reported in Tables 2 and 3 with 4 and 5 shows that the brightness transfer function can yield a more robust matching. That is because these functions try to map similar colors in two camera views to each other. Consequently, by using properly estimated functions, the overlap of correspondent codewords increases and yields a better match. Another fact to note is the resemblance of Tables 3 and 5. That is because the problem of illumination changes is less serious in the second scenario. It can be inferred from Figure 7 and especially Figure 8 with identity like functions in the right column.

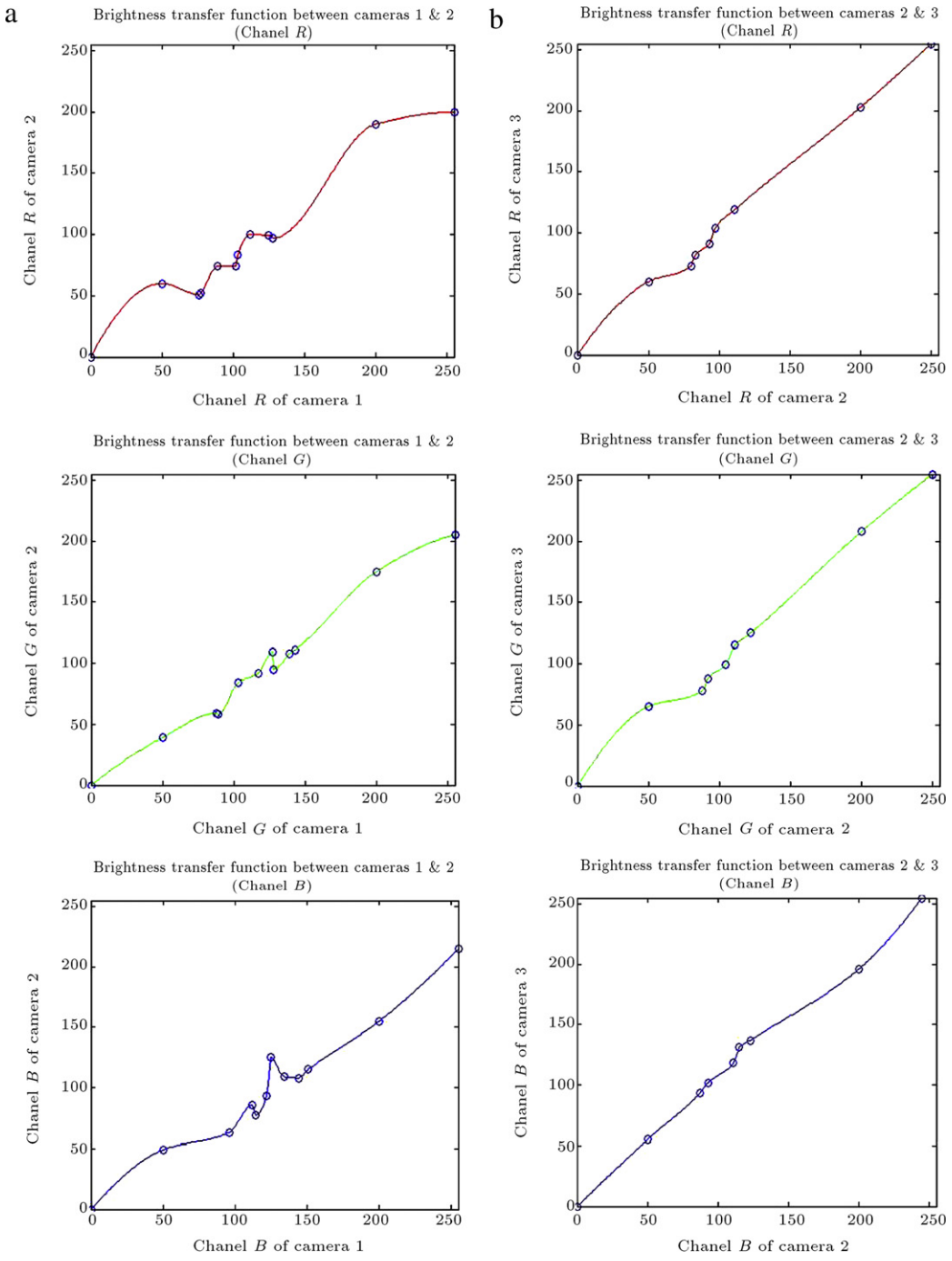

Figure 8: Brightness transfer function between: (a) Camera 1 & 2, (b) Camera 2 & 3. Rows 1-3 are for RGB channels.

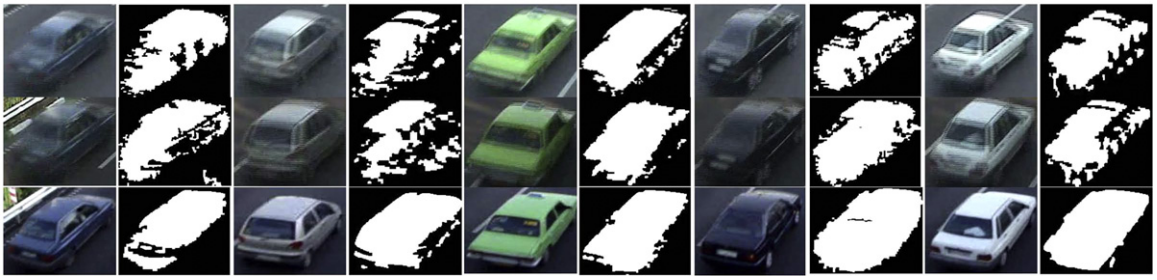

Figure 9: A sample view of five selected vehicles for our evaluation purpose in three cameras. Rows 1–3 for Camera 1–3 views.

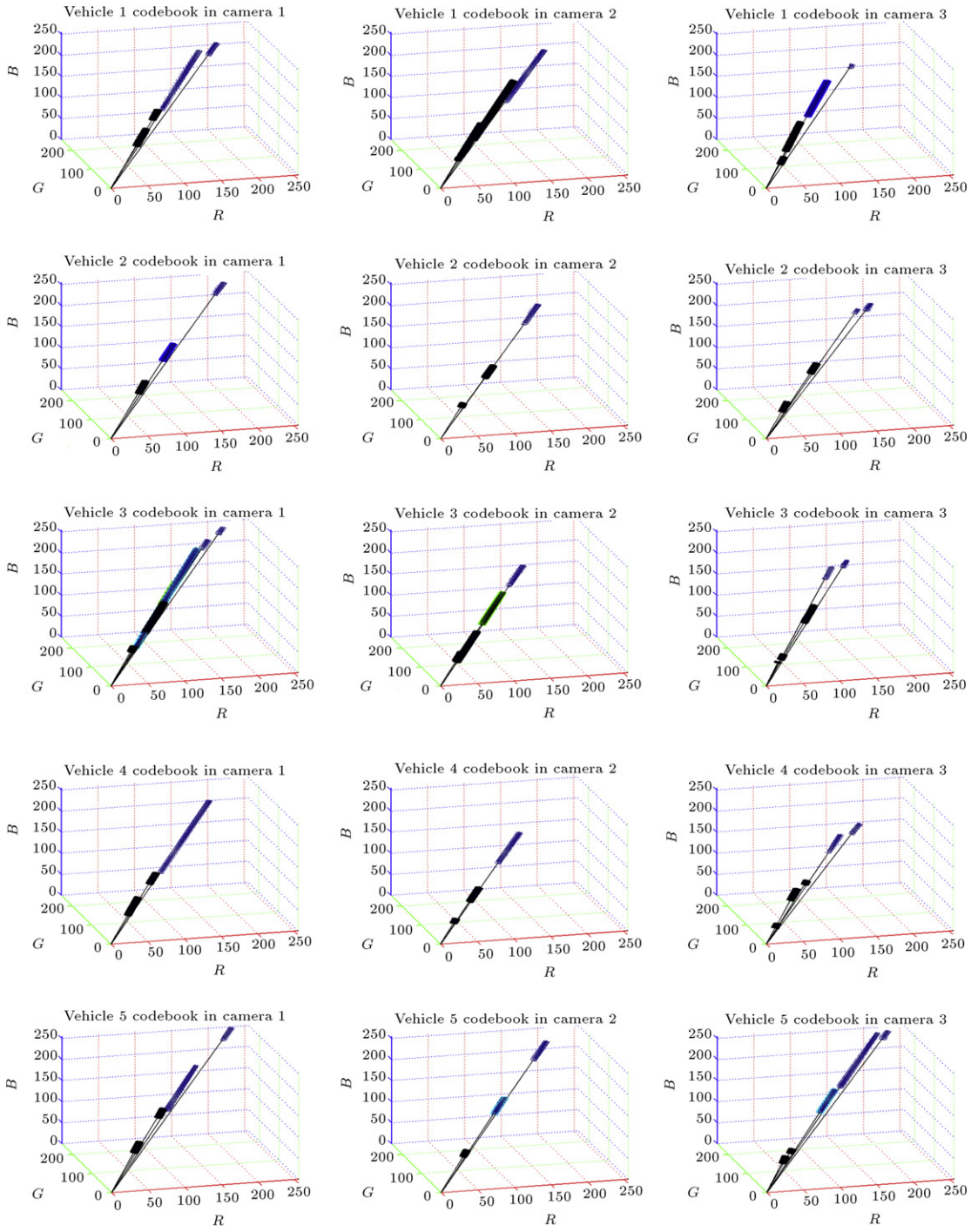

Figure 10: Codebook of five selected vehicles in three cameras. Rows 1–5 are for vehicles 1–5. Columns 1–3 are for Cameras 1–3.

5. Conclusion

In this paper, a novel codebook representation method for appearance modeling of moving vehicles was presented. The modeling could cope with illumination changes of environment. It also was able to accommodate the different viewing angles of objects. In this model, instead of keeping the histogram bins of each object, the major colors of that object were preserved by using the related cylinders in the RGB space. Since these cylinders can grow along the illumination axis, the method could keep the track of major colors in the interest field of view of each camera. To measure the similarity between obtained codebooks of objects, learned transfer functions were used to map the cylinders to their most similar one. Although this modeling was presented for vehicles, it has the capability of being used for any other object with limited number of color clusters (*e.g.*, pedestrians), which is the topic of our next future work.

References

[1] Hu, W., Wang, T. and Maybank, S. "A survey on visual surveillance of object motion and behaviors", *IEEE Transactions on Methods, Man, Cybernetic*, 34(3), pp. 334–352 (2004).

[2] Moeslund, T., Hilton, A. and Kruger, V. "A survey of advances in vision-based human motion capture and analysis", In *Int. J. Comput. Vision and Image Understanding*, pp. 90–126, Elsevier Science Inc, New York, NY, USA (2006).

[3] Valera, M. and Velastin, S. "Intelligent distributed surveillance methods: a review", *IEEE Proc. Vision Image Signal Processing*, 152(2), pp. 192–204 (2005).

[4] Stauffer, C. and Grimson, W. "Adaptive background mixture models for real-time tracking", *IEEE Computer Society Conf. on Computer Vision and Pattern Recognition*, pp. 246–252 Fort Collins, CO, USA (1999).

[5] Yilmaz, A., Javed, O. and Shah, M. "Object tracking: a survey", *ACM Computing Surveys (CSUR)*, 38(4), pp. 13–58 (2006).

[6] Streit, R.L. and Luginbuhl, T.E. "Maximum likelihood method for probabilistic multihypothesis tracking", *Proceedings of the Int. Society for Optical Engineering (SPIE)*, 2235, pp. 394–405 (1994).

[7] Comaniciu, D., Ramesh, V. and Andmeer, P. "kernel-based object tracking", *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 25, pp. 564–575 (2003).

[8] Kang, J., Cohen, I. and Medioni, G. "Continuous tracking within and across camera streams", *IEEE Conf. Computer Vision Pattern Recognition* (2003).

[9] Khan, S. and Shah, M. "Consistent labeling of tracked objects in multiple cameras with overlapping fields of view", *IEEE Transaction on Pattern Analysis Mach. Intell*, 25(10), pp. 1355–1360 (2003).

[10] Mittal, A. and Davis, L.S. "M2 tracker: a multi-view approach to segmenting and tracking people in a cluttered scene", *Internation Journal of Computer Vision*, 51(3), pp. 189–203 (2003).

[11] Shan, Y., Sahwney, H.S. and Kumar, R. "Unsupervised learning of discriminative edge measures for vehicle matching between nonoverlapping cameras", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(4), pp. 700–711 (2008).

[12] Madden, C., Cheng, E.D. and Piccardi, M. "Tracking people across disjoint camera views by an illumination-tolerant appearance representation", *Machine Vision and Applications*, 18, pp. 233–247 (2007).

[13] Huang, T. and Russell, S. "Object identification in a bayesian context", *Proceedings of IJCAI* (1997).

[14] Javed, O., Shafique, K., Rasheed, Z. and Shah, M. "Modeling inter-camera space–time and appearance relationships for tracking across nonoverlapping views", In *Computer Vision and Image Understanding, 109*, pp. 146–162, Elsevier Science Inc., New York. NY. USA (2008).

[15] Shabaninia, E. and Kasaei, Sh. "Neighboring Vehicles Modeling for Tracking across Nonoverlapping Cameras", *Iranian Conf. Electrical Engineering ICEE2010*, Isfahan, Iran, 2010.

[16] Kettnaker, V. and Zabih, R. "Bayesian multi-camera surveillance", *IEEE Conf. Computer Vision Pattern Recognition*, pp. 117–123 (1999).

[17] Chen, K., Lai, Ch., Hung, Y. and Chen, Ch. "An adaptive learning method for target tracking across multiple cameras", *IEEE Conf. on Computer Vision and Pattern Recognition* (2008).

[18] Teixeira, L.F. and Corte-Real, L. "Video object matching across multiple independent views using local descriptors and adaptive learning", In *Pattern Recognition, 30(2)*, pp. 157–167, Elsevier Science Inc., New York, NY, USA (2009).

[19] Makris, D., Ellis, T.J. and Black, J.K. "Bridging the gaps between cameras", *IEEE Conf. Computer Vision and Pattern Recognition* (2004).

[20] Shabaninia, E. and Kasaei, Sh. "A novel vehicle tracking method with occlusion handling using longest common substring of chain-codes", *Int. CSI Conf. (CSICC2009)*, Tehran, Iran 2009.

[21] Li, L., Huang, W., Gu, I.Y.H., Leman, K. and Tian, Q. "Principal color representation for tracking persons", *Proceedings of SMC*, 1, pp. 1007–1012 (2003).

[22] Spath, H., *Cluster Analysis Algorithms*, Ellis Horwood Ltd, Chichester, U.K. (1980).

[23] Kim, K., Chalidabhongse, H., Harwood, D. and Davis, L. "Real time foreground–background segmentation using codebook model", *Real–Time Imaging*, pp. 172–185 (2005).

[24] Prosser, B., Gong, S. and Xiang, T. "Multi-camera Matching Using Bi-Directional Cumulative Brightness Transfer Functions", *Proceedings of British Machine Vision Conf.*, Leeds (2008).

[25] Prosser, B., Gong, S. and Xiang, T. "Multi-camera matching under illumination change over time", *Proceedings of ECCV Workshop on Multi-camera and Multi-modal Sensor Fusion Algorithms and Applications*, Marseille, France (2008).

Elham Shabaninia was born in Iran, in 1984. She received the B.S. and M.S. degrees both in Computer Engineering from Shahid Bahonar University of Kerman and Sharif University of Technology of Tehran in 2006 and 2009, respectively. She is currently working as a lecturer in the Department of Computer Engineering at Shahid Bahonar University of Kerman. Her research interests are object tracking, machine vision and video processing.

Shohreh Kasaei received the B.Sc. degree from the Department of Electronics, Faculty of Computer and Electrical Engineering, Isfahan University of Technology (IUT), Iran, in 1986. She worked as research assistance in Amirkabir University of Technology (AUT), for three years. She then received the M.Sc. degree from the Graduate School of Engineering, Department of Electrical and Electronic Engineering, University of the Ryukyus, Japan, in 1994, and the Ph.D. degree from Signal Processing Research Centre (SPRC), School of Electrical and Electronic Systems Engineering (EESE), Queensland University of Technology (QUT), Australia, in 1998. She was awarded as the best graduate student in engineering faculties of University of the Ryukyus, in 1994, the best Ph.D. student studied in overseas by the ministry of Science, Research, and Technology of Iran, in 1998, and as a distinguished researcher of Sharif University of Technology (SUT), in 2002 and 2010, where she is currently a professor. She is the director of Image Processing Lab (IPL) at Sharif University of Technology. Her research interests are in image processing with primary emphasis on multi-resolution texture analysis, 3D computer vision, 3D object tracking, 3D model building, scalable video coding, image retrieval, video indexing, face recognition, hyperspectral change detection, video restoration, fingerprint authentication, and watermarking.